

# **HandyBot™**

**(Universal Mobile Robot platform)**

## **Operation Manual V2.0**

### **IMPORTANT!!**

**It is highly recommended to read this manual before working on HandyBot™ kit.**

This document is Copyright © 2000–2001 by Robot Store (HK). It may be distributed freely in verbatim form provided that no fee is collected for its distribution and this copyright notice is included. An electronic version of this document is available from the Robot Store (HK) homepage at <http://www.robotstorehk.com/>.

<b>Revision History</b>		
<b>Document version:</b>	<b>Descriptions:</b>	<b>Date Modified:</b>
0.0	Initial Release	Feb 15, 2001
1.0	Insert motor specifications	Mar 20, 2001
1.1	Minor Typo error correction	May 30, 2001
1.2	Example source code added	Sept 20, 2001
2.0	Minor changes on HandyBot™ part list	Dec 12, 2001

## Table of Contents

Table of Contents.....	3
1.0 Introduction .....	4
2.0 HandyBot™ Quick Start-up Guide .....	6
2.1 Battery Installation.....	6
2.2 Interactive-C 3.x Software Installation.....	18
2.3 Downloading Interactive-C Pseudo-Code (Pcode) into HandyBot™ .....	19
2.4 Expansion Board Software Installation (optional) .....	22
3.0 Your first program – “Robot, let’s have a dance...” .....	24
3.1 Loading the program “dance.c” .....	24
3.2 What did you learn form the program “dance.c?” .....	29
4.0 How to interface the provided sensor? .....	31
5.0 FAQ .....	31
6.0 Support and Contact Information .....	32

## 1.0 Introduction

Thank you for choosing HandyBot™ -- universal mobile robot base platform dedicated to robotic experiments!

HandyBot™ is a complete mobile robot base using a well-known MIT Handyboard system as its brain. It has two high quality planetary gear-head motors (manufactured by COPAL in Japan) and a caster for locomotion. The MIT Handyboard is placed on top of large PCB fabric (91mm x 140mm) platform for easy mounting of various sensors. This ensures the platform has the best performance to meet your specific requirements.

### HandyBot™ Technical Specifications:

<b>Platform size (with Handyboard system):</b>	91mm (W) x 140mm (L) x 120mm (H) PCB fabric for easy mounting of sensors
<b>Wheels diameter:</b>	56mm
<b>Front wheel:</b>	free rolling caster
<b>Driven system:</b>	Differential drive
<b>Motors:</b>	2 DC 6v-9v COPAL Planetary gear-head motors
<b>Handyboard power supply:</b>	DC 10v-15v (8 pcs x AA size battery for long running time). Alkaline or rechargeable battery.
<b>Motor power supply:</b>	DC 9v (6 pcs x AA size battery for long running time). Alkaline or rechargeable battery.

### HandyBot™ Parts List:

<b>Part Descriptions:</b>	<b>Quantity</b>
MIT Handyboard system	1
Serial interface board	1
DB9 serial cable	1
RJ11 cable	1
Handyboard Expansion Board (optional)	1
Interactive-C 3.x disk (software and Interactive-C manual)	1
Handyboard manual disk	1
HandyBot™ Example Source Code disk	1
HandyBot™ with 2 motors and wheels	1
8 pcs x AA size battery holder	1
6 pcs x AA size battery holder	1
16 pcs AA size alkaline battery	1
Bumper (Microswitch)	6
Photo-resistor (CDS)	4
Beam-breaker	2
IR transmitter LED	2

IR Receiver LED	2
Spacer with screws	4
Screws and nuts	10
Double side foam tape (to mount sensors)	2

The HandyBot™ package includes various sensors. The designs of sensor interface circuits are left to the user. For a reference design, please consult “Handyboard Technical Reference Manual” document provided with the kit. Also, a good Handyboard “bible” – Robotic Explorations -- written by Dr. Fred Martin has full of Interactive-C (IC) source code and hardware interface examples.

Note: Dr. Fred Martin’s book used MIT Handyboard system and LEGO parts to build the robots. However, the building concepts are the same, and both hardware and software can be applied on HandyBot™ without making any major changes.

It is highly recommended to read section 2.0 “HandyBot™ Quick Start-up Guide” before proceeding.

**WARNING: Turn-off power before adding any sensor or hardware to the HandyBot™. Otherwise, the board will be damaged.**

## 2.0 HandyBot™ Quick Start-up Guide

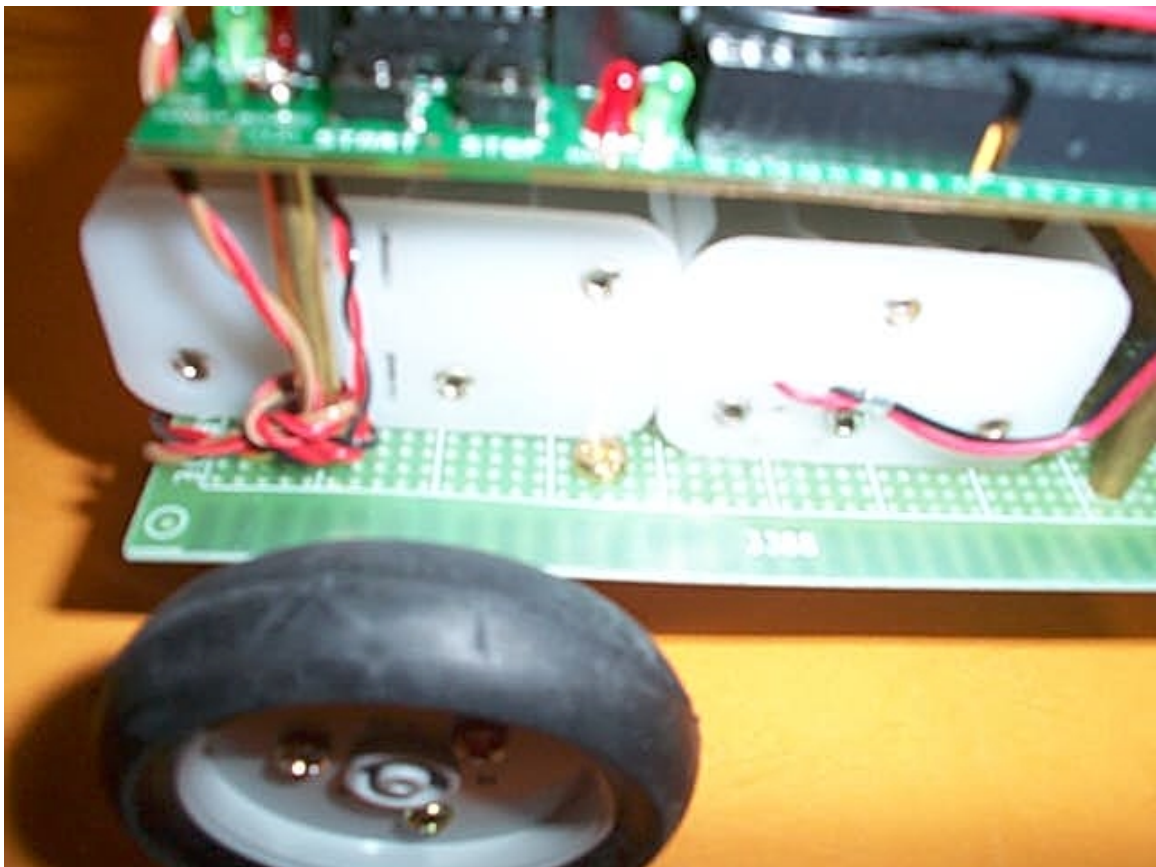
This section will show you how to power up the HandyBot™ and write a simple IC program to “play” with it.

Underneath the MIT Handyboard, you will find two empty battery holders. One is 8 x AA battery holder for Handyboard main power supply. Another is 6 x AA battery holder for motors power supply.

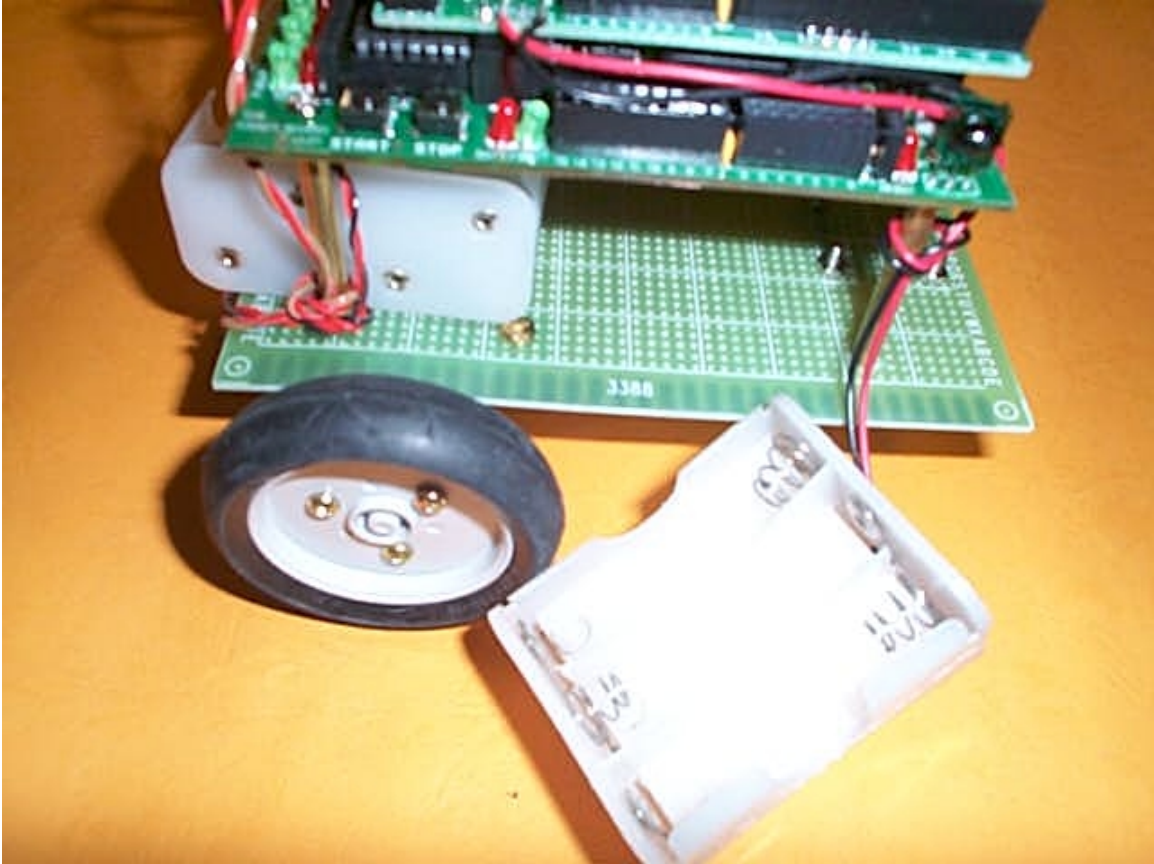
### 2.1 Battery Installation

There is a trick to remove and insert the battery holders to HandyBot™. Please follow the instructions below.

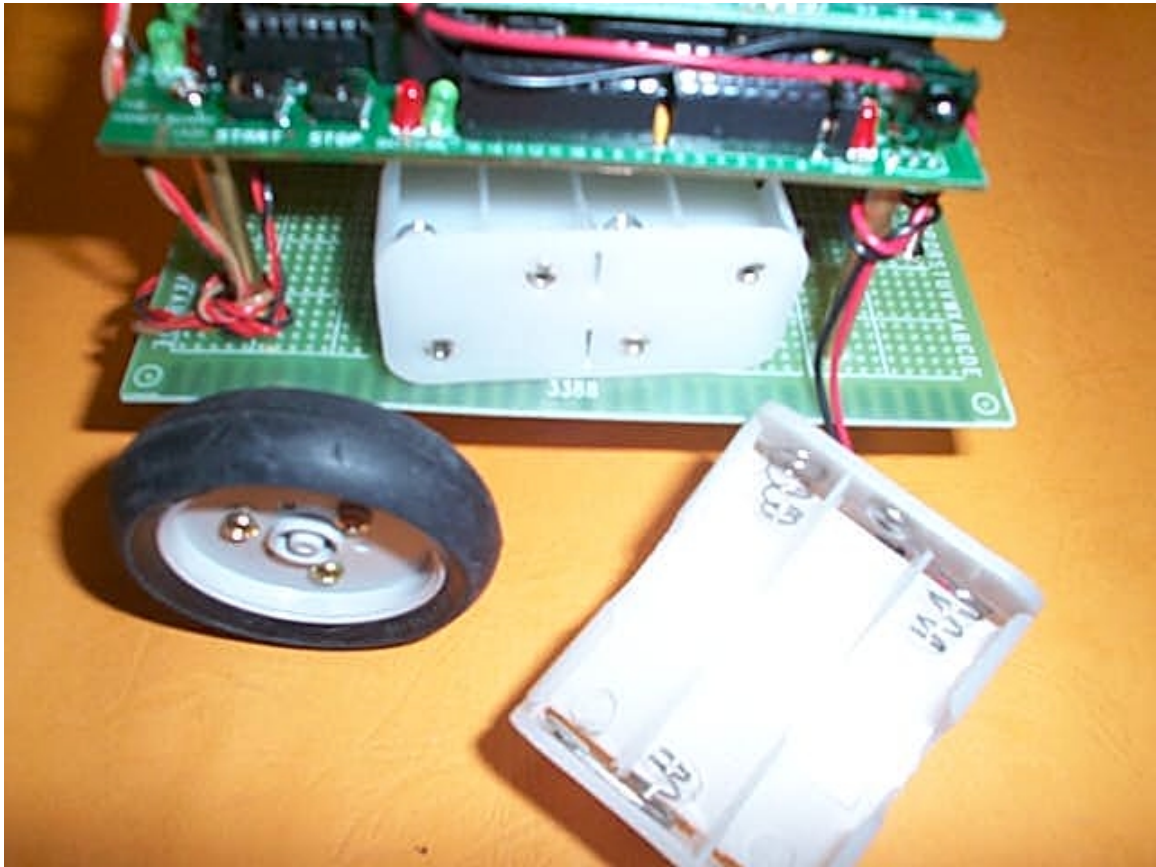
Push back 8xAA battery holder a little bit to left some space.



Taking out 6xAA battery holder from the space left.

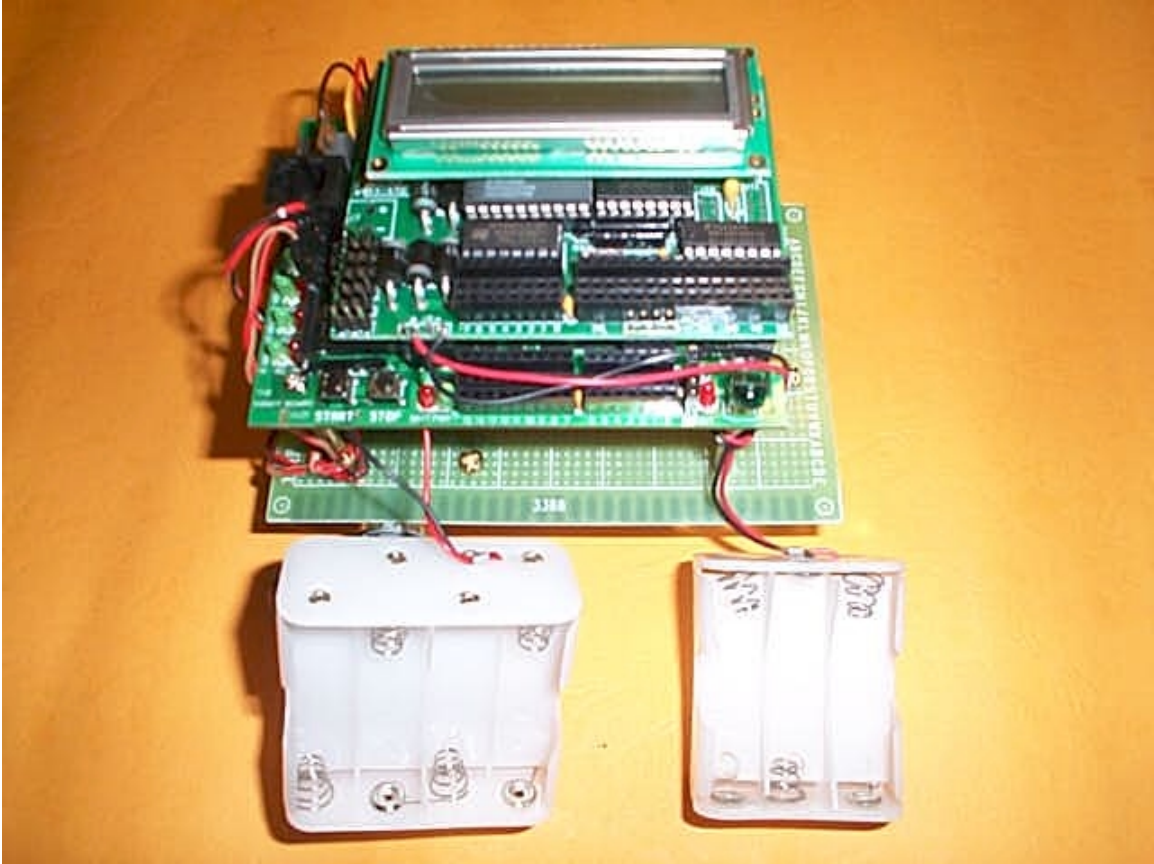


Then, you can take the 8 x AA battery holder out of the platform.



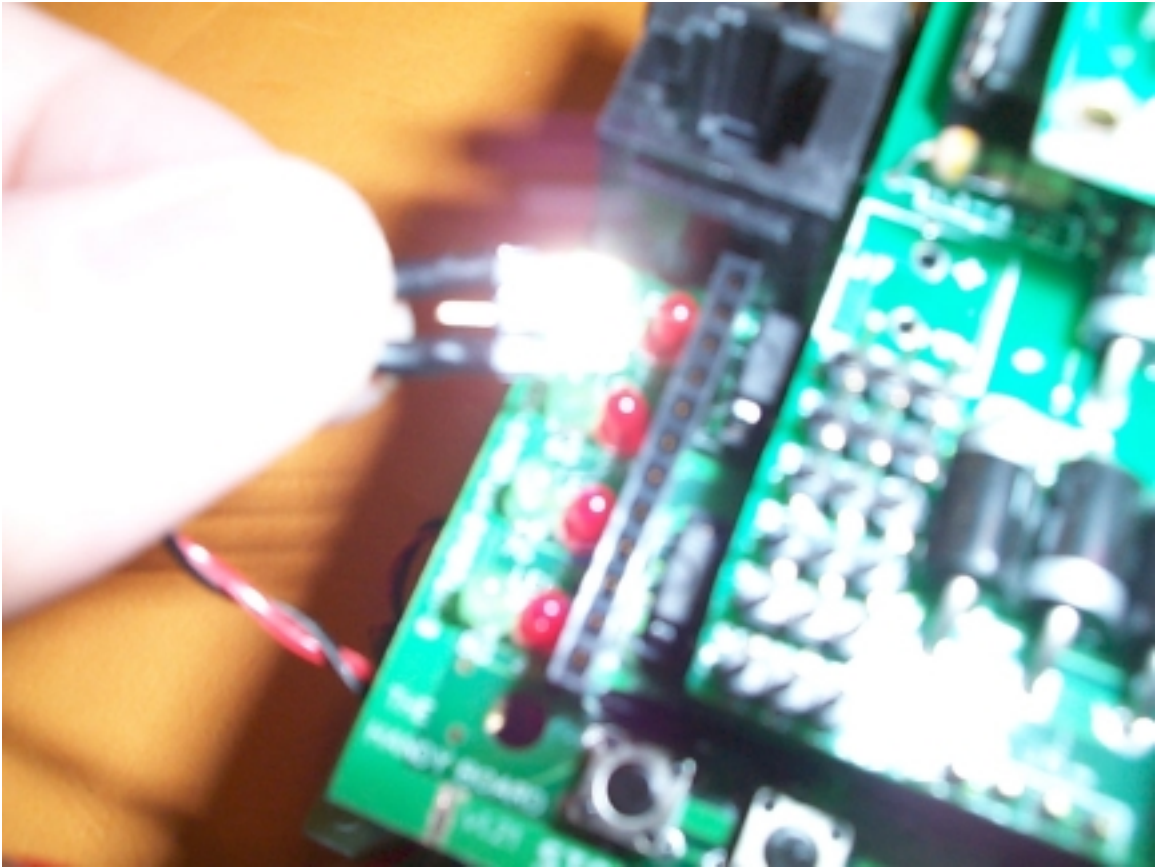


You are ready to place the batteries on HandyBot™.



**WARNING:** Due to the unknown state when HandyBot™ starting up, please remove any motor connectors from the HandyBot™ before inserting the batteries. Otherwise, the motors may turn on suddenly.

Remove two motor connectors from the Handyboard

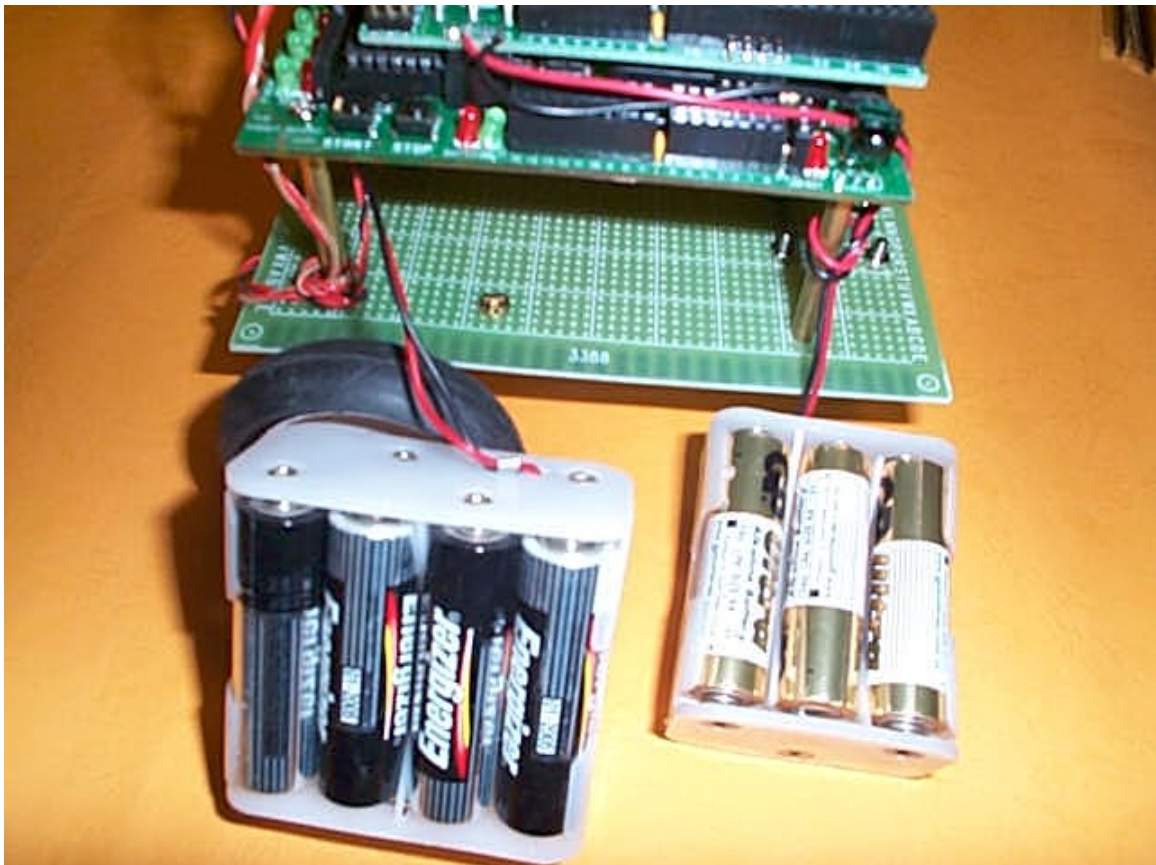


Now, you are ready to insert alkaline batteries to the battery holders. Besides alkaline type batteries, you can use any rechargeable batteries to power up HandyBot™.

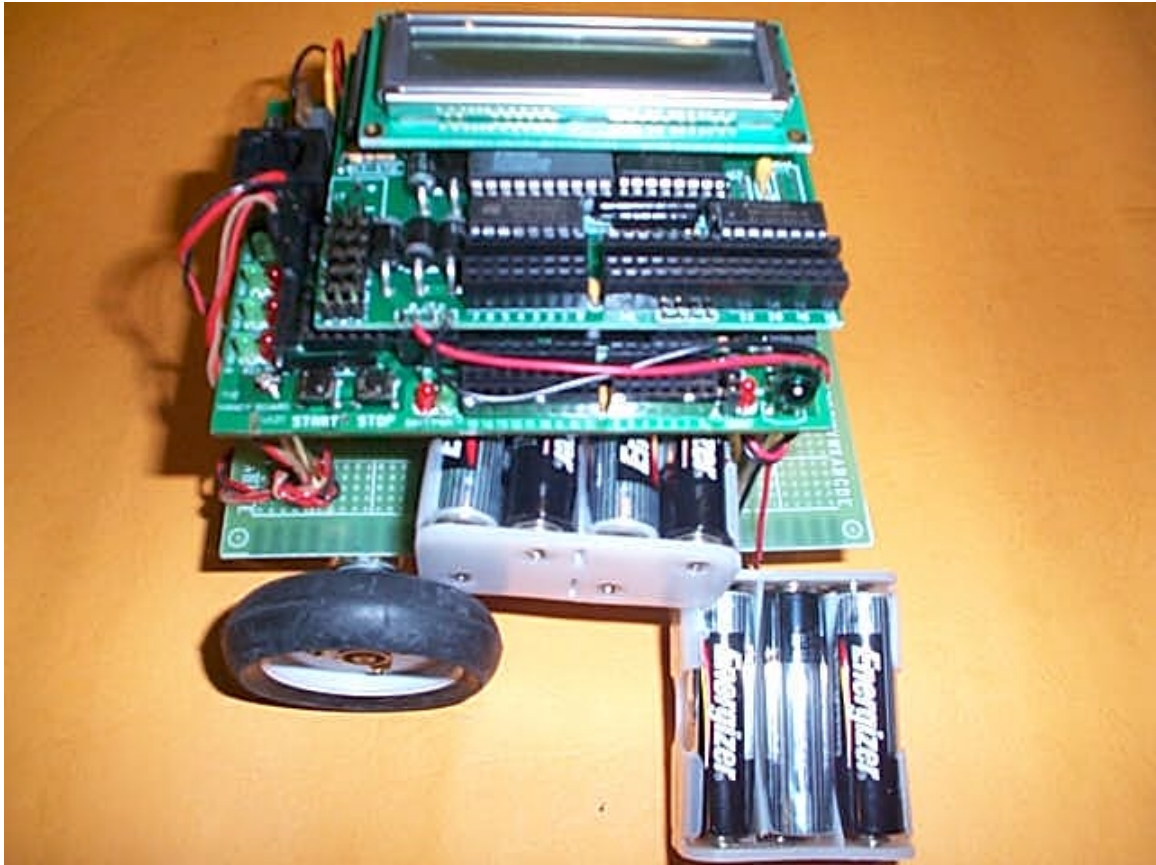
**WARNING: Alkaline or rechargeable batteries can be used on HandyBot™.**

Place 8 pieces alkaline batteries (provided with the kit) into the 8xAA battery holder and 6 pieces alkaline batteries (provided with the kit) into the 6xAA battery holder.

Note: The Handyboard main power supply (8xAA size) is connected to the Handyboard main board while motor power supply (6xAA size) is connected to J6 of Handyboard Expansion board.

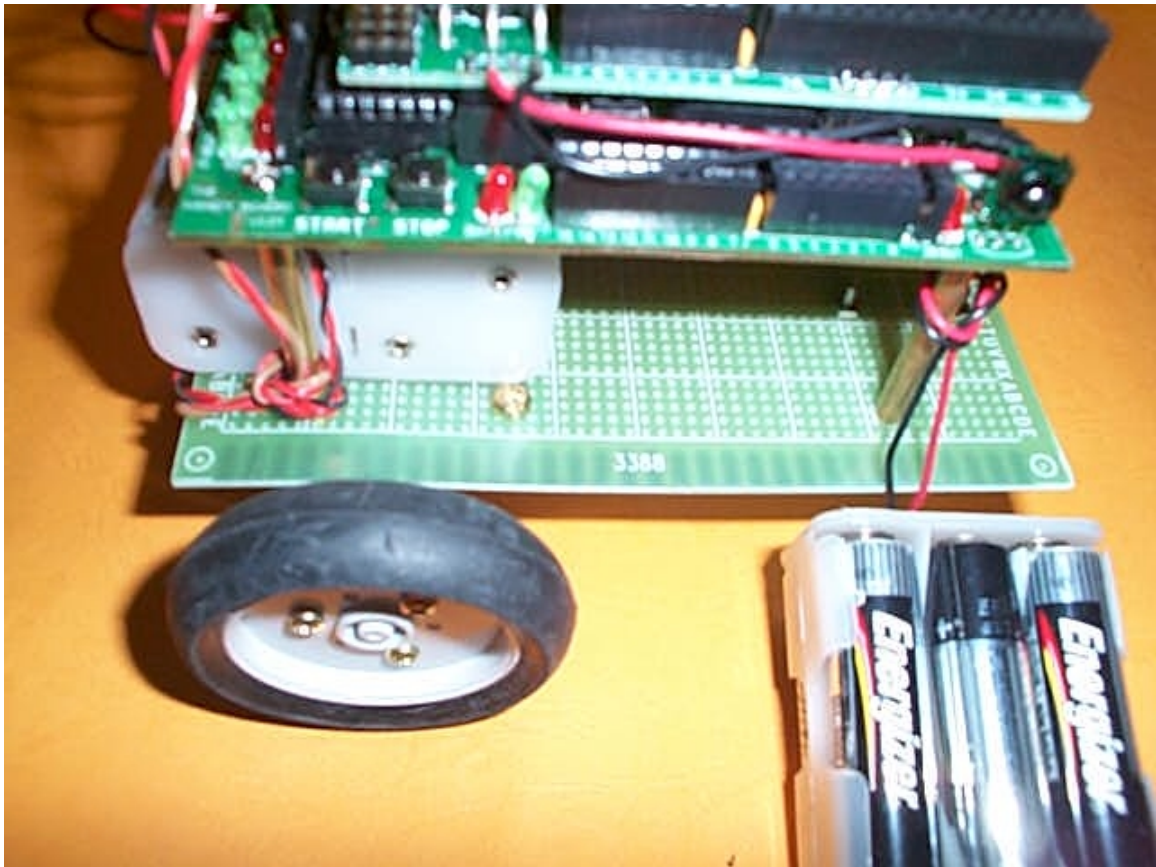


Insert back the 8xAA battery holder to the HandyBot™





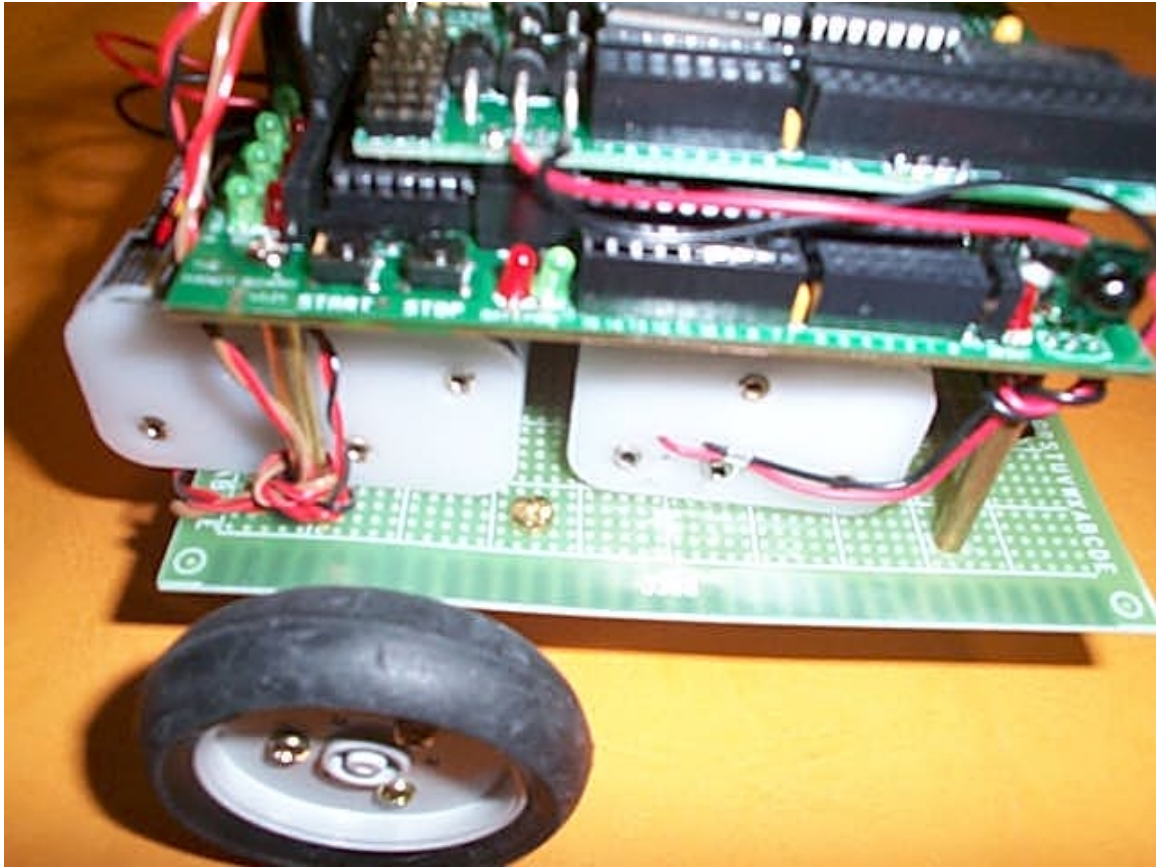
Push back the 8xAA battery holder a little bit.



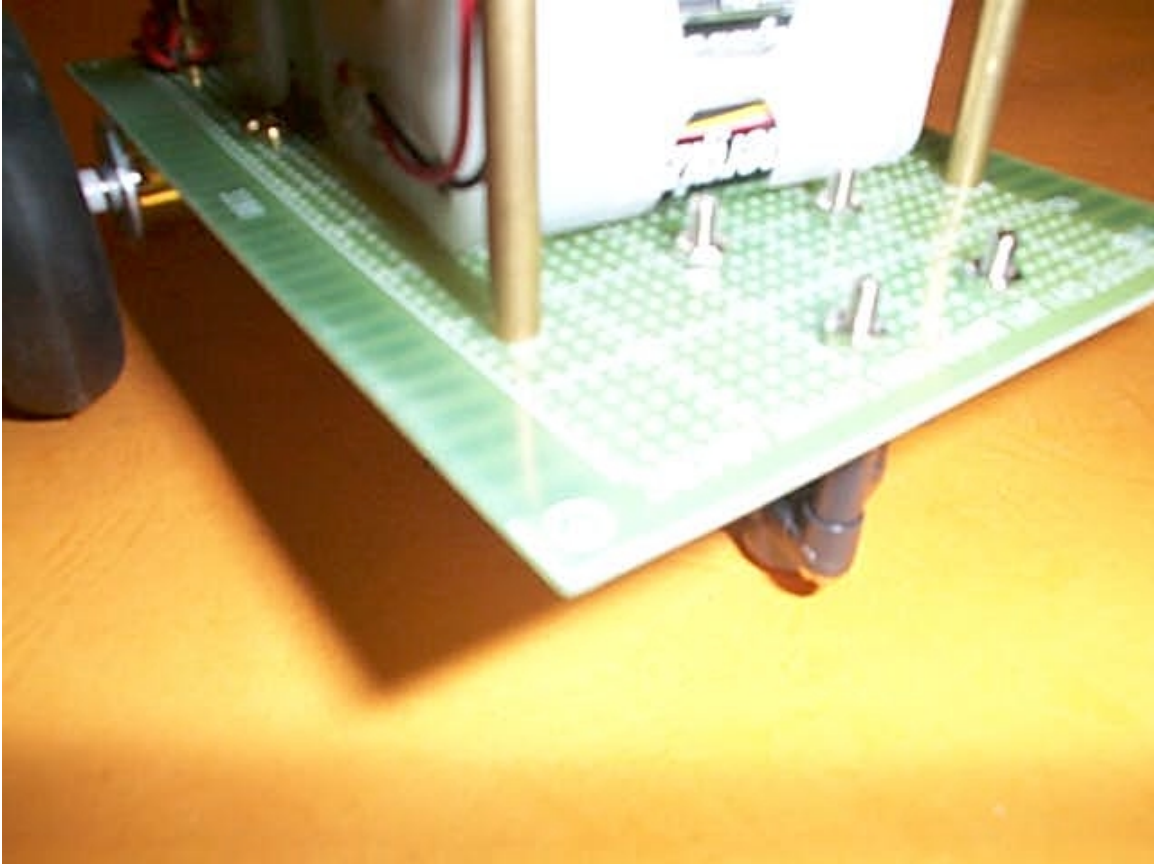
Now, it is ready to insert back the 6xAA-battery holder



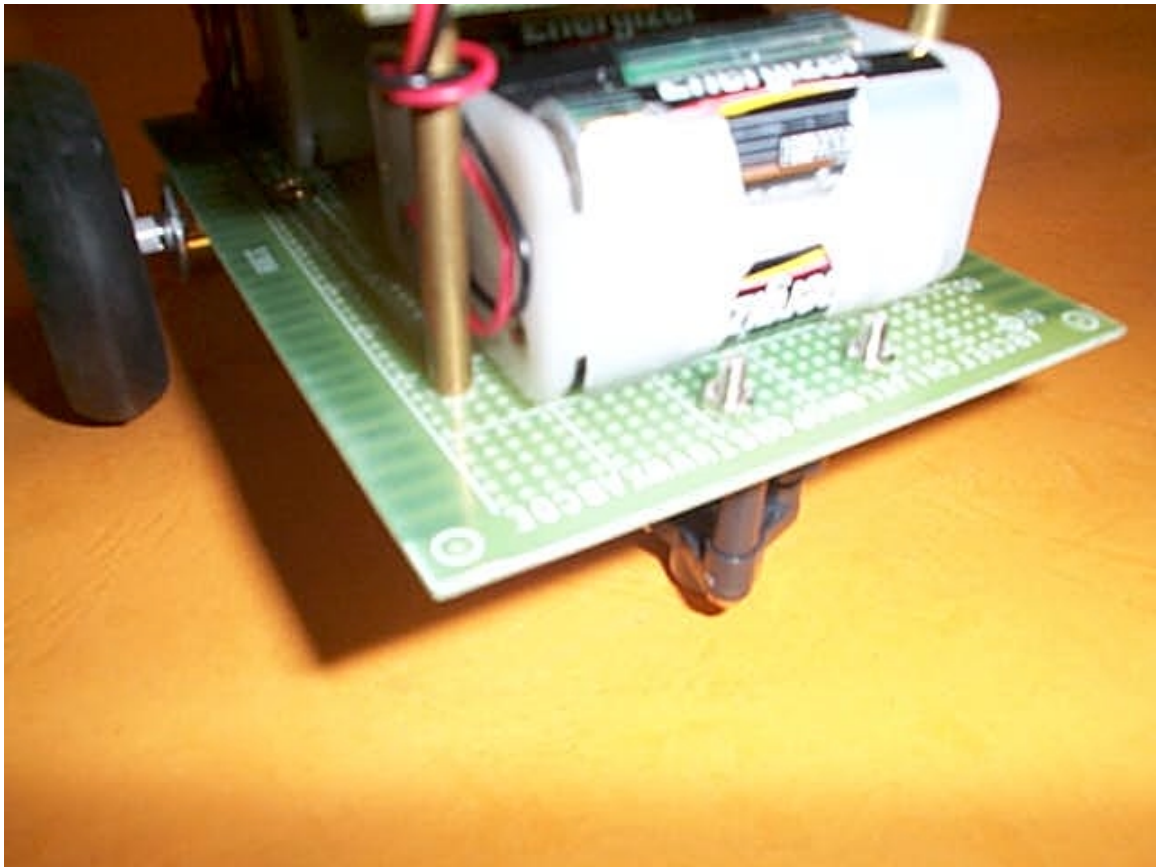
Almost done.



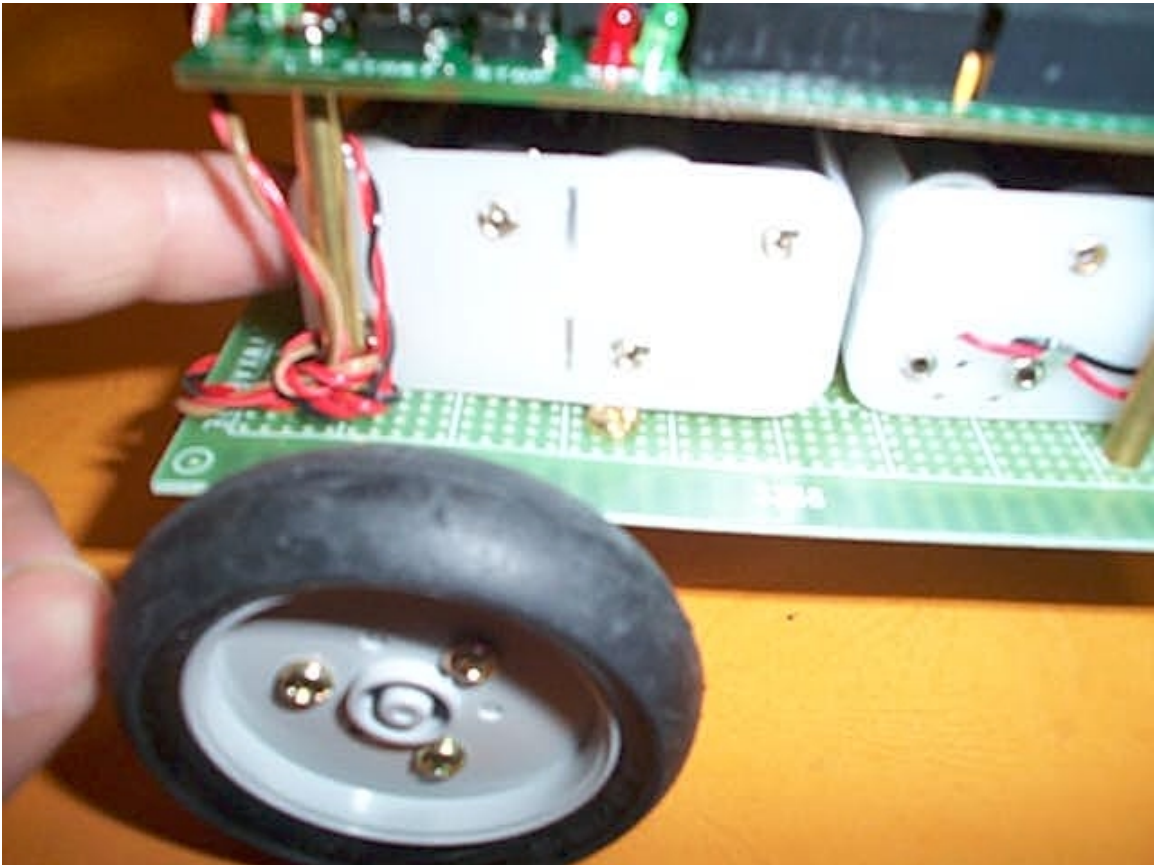
Push the 6xAA battery holder a little bit front of the HandyBot™. Two screws attached on the caster will be under 6xAA battery holder.







Place the 8xAA battery holder close to the center of platform. It is done.  
Reverse the above procedures if removing battery from the battery holders.



## **2.2 Interactive-C 3.x Software Installation**

The HandyBot™ comes with an Interactive-C 3.x under WINDOWS 9x/NT platform. You should find a disk labeled “Interactive-C 3.x for WINDOWS 9x/NT.”

Interactive-C is an ANSI-C like programming language and is dedicated for robotic programming. It comes with useful robotic control libraries such as DC motor control, servos control, LCD printing functions, music player, digital and analog sensor drivers.

To fully utilize the programming features of Interactive-C, please refer to the user manual called “icmanual.pdf” located on the same disk

Run the setup program “ICW32.EXE” on the disk and install it into your local hard disk. The installation default directory is C:\IC and the Interactive-C libraries are located in C:\IC\LIBS.

Go to Windows Start->Programs->Interactive-C->Interactive-C to execute Interactive-C. It is required to enter Registration Key when the Interactive-C is first being activated.

Browse the file called “key.txt” from the same disk. Paste the Licensee and Registration Key on the Interactive-C registration key request screen.

If the key is correct, the IC will try to communicate with HandyBot™. You will see "**Board not responding, Config board settings**", press "NO" and exit IC since we will run it all over again once HandyBot™ connected.

### **2.3 Downloading Interactive-C Pseudo-Code (Pcode) into HandyBot™**

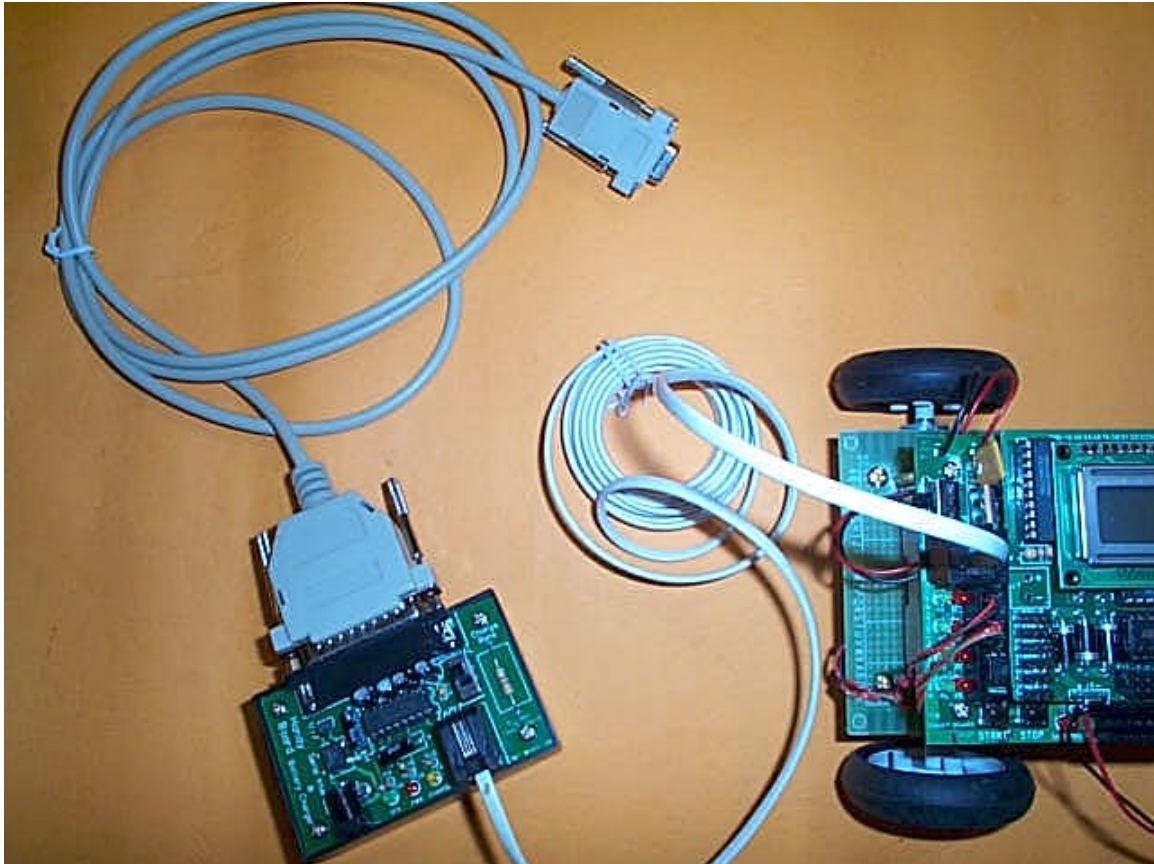
To run the Interactive-C on HandyBot™, the Pseudo-Code (Pcode) must be loaded before downloading any program. The Pcode is a C interpreter, which is loaded into HandyBot™. Your program will be interpreted during run-time. That's why you can load, unload, and execute your program right away.

**WARNING: Due to the unknown state when HandyBot™ starting up, please remove any motor connectors from the HandyBot™ before inserting the batteries. Otherwise, the motors may turn on suddenly.**

Here are procedures to load the Pcode into HandyBot™:

1./ Your HandyBot™ is connected to Serial interface board via RJ11 and the Serial interface board is then connected to PC serial port (DB9). Turn on the power of the HandyBot™. Power switch is located on Handyboard main board and nearby RJ11 socket. **Again, make sure the motors are not connected to Handyboard!!**

The DB9 of the serial cable should connect to DB9 serial port of PC (not shown).



2./ Run the IC3.x on your PC.

3./ You will see "Board not responding, Config board settings", press "YES"

4./ Select appropriate COM port and press "DOWNLOAD PCODE"

If you are not sure which COM, you can try it one by one for the whole bootstrap steps.

5./ Open "Handy\_Board\_1.2"

6./ Set the Handyboard to the Bootstrap Mode. You should follow the instructions given

by the Interactive-C message box to see how to set the board to the bootstrap mode. Both RED and GREEN LEDs nearby the STOP button should be OFF. The instruction reads:

"To put board into download mode, hold down STOP button while turning on the board. Continue holding STOP button until the green PWR LED goes out."

The STOP button is located in Handyboard. You can see the STOP label.

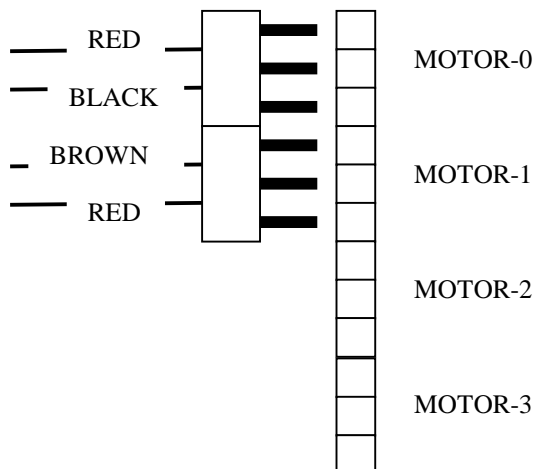
7./ Press OK on WINDOWS prompt, the GREEN nearby the STOP button will turn on and blinking for a while. Pcode is being downloaded.

8./ When it is done, the IC will prompt you to turn off and on the power of Handyboard again and press OK on IC prompt to load Pcode. You will hear a beep sound and a message on the LCD display will be shown: "**IC v3.1 Handy Board 1.2**" The Pcode is loaded. Leave the HandyBot™ power on for the next section.

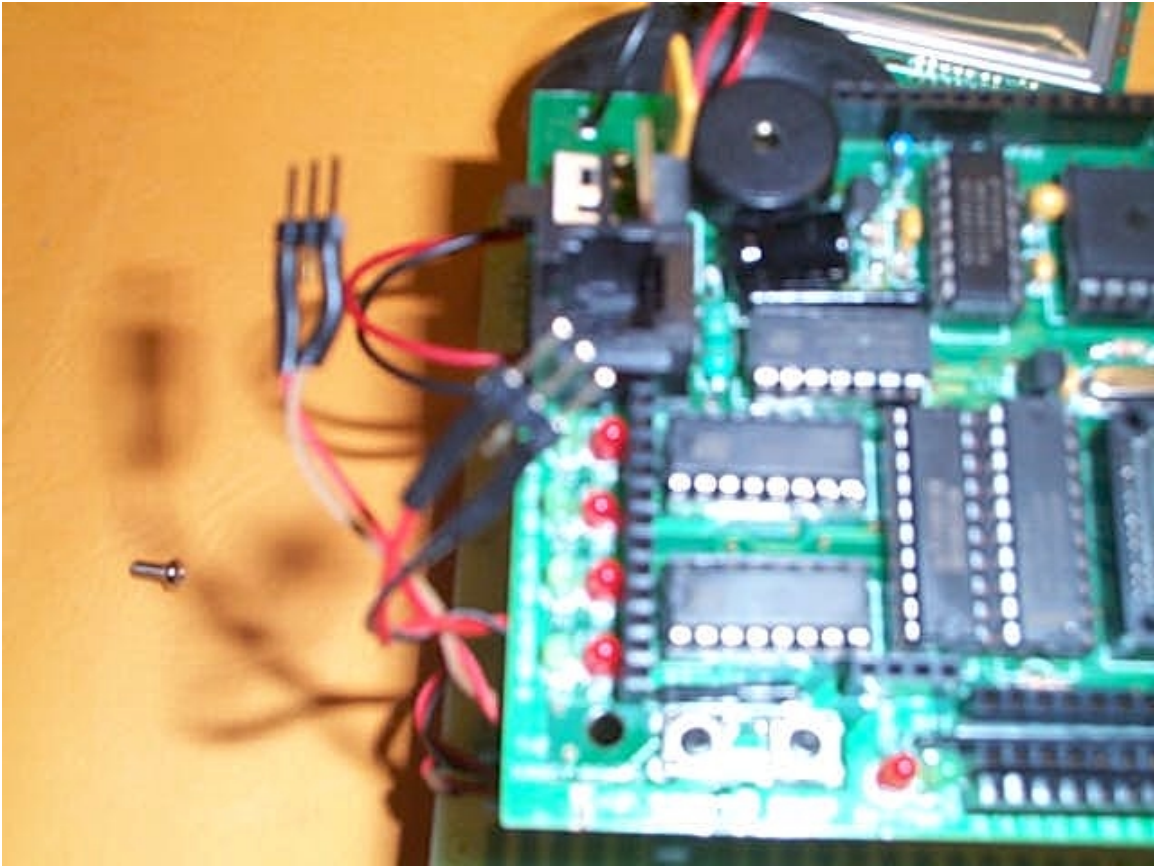
Insert the motor connectors back to Handyboard motor port locations: MOTOR-0 and MOTOR-1. MOTOR-0 port is nearby RJ11 socket. You can read the silk screen on Handyboard. MOTOR-0 and MOTOR-1 will be used for the HandyBot™ example test code below. There is no polarity on DC motor, but the polarity will affect the motor rotate direction.

To have a smooth performance on the next section example code, please follow the following connection:

Motor Connector on Handyboard







## 2.4 Expansion Board Software Installation (optional)

By using Handyboard Expansion, you will get more digital and analog inputs, driving more servos, dedicated LEGO motors interface, and Polaroid sonar interface. Unzip the HBEXP.ZIP from the disk 2: "Interactive-C 3.x for WIN9x/NT" to C:\IC\LIBS. Overwrite the existed files. Follow the instructions below:

### EXPANSION BOARD LIBRARIES README

Please put all enclosed files in your Interactive C library folder. This is typically C:\IC\LIBS\. The IC application uses a hard-coded path, so all library files must be put in this directory.

Please note that the enclosed "lib\_hb.c" *\*replaces\** the old one. When you are not using the Expansion Board, the "hbsensor.c" file should be loaded *\*instead\** of "expensens.c". Modify your lib\_hb.lis to include hbsensor.c instead of expensens.c.

You may find Expansion board library function call reference in the document “Expansion Board Software” on the same disk.

### 3.0 Your first program – “Robot, let’s have a dance...”

At this point, you are successfully installed the battery to HandyBot™ and Interactive-C software into your PC, and also you can boot-up the Handyboard system. Am I correct? You can see the beating heart on HandyBot™ LCD screen.

It is now your turn to use your imagination to develop robotic control algorithm programs. Before you start programming your own, here is an example illustrates few concepts you may ask latter in the development.

You will find all example source code in the disk 3 labeled as “HandyBot™ Example Source Code.” To perform the experiments, please copy all the HandyBot™ example source code files to C:\IC\LIBS.

Please make sure you have 1.5m x 1.5m free space to play HandyBot™ with the following program. Here is your first program “dance.c” – illustrated below.

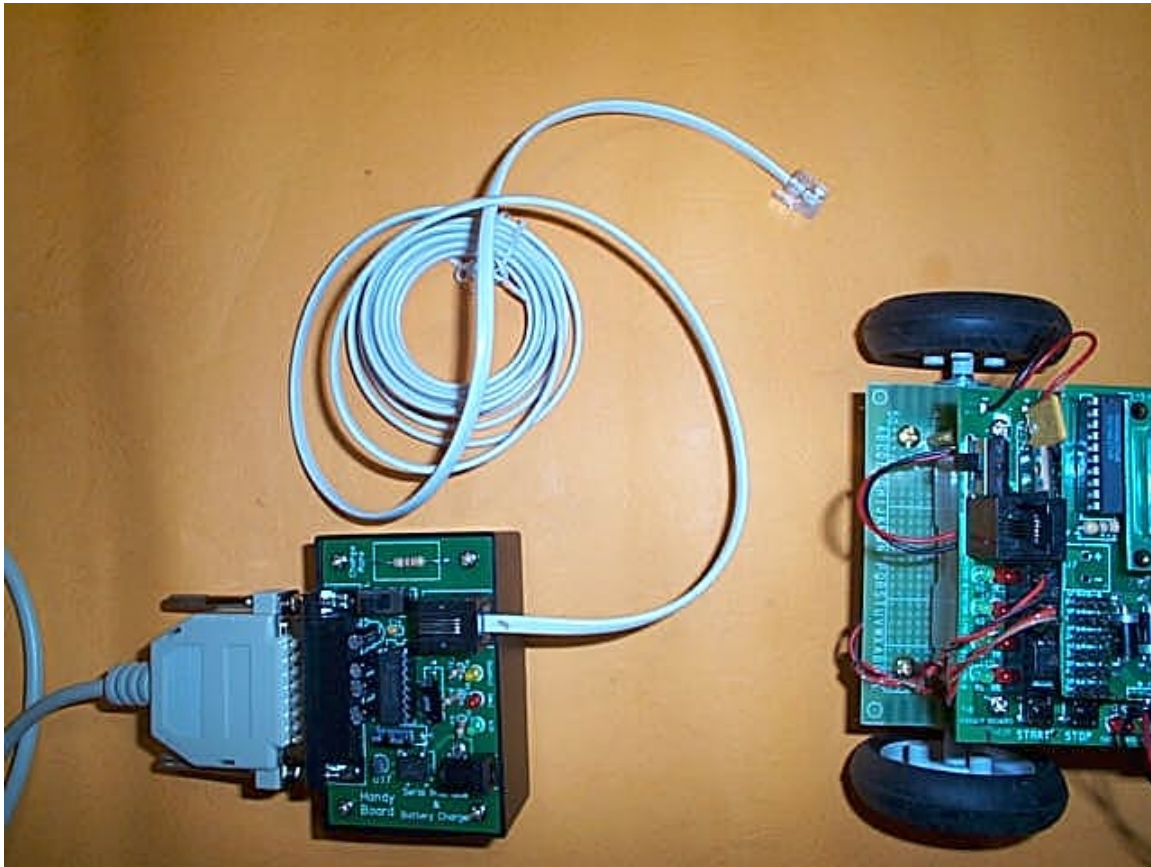
#### 3.1 Loading the program “dance.c”

If you are reading this manual in order, your HandyBot™ is now connected to your PC, and the Pcode is loaded into Handyboard already. Your PC should have Interactive-C prompt, and the LCD on Handyboard shows “**IC v3.1 Handy Board 1.2**”

If this is the case, you are ready to load the program into Handyboard. From the PC Interactive-C prompt, type in “**load dance.c**” The Interactive-C will start loading “dance.c” from your default working directory -- C:\IC\LIBS to the HandyBot™.



If the download is successful, **turn off HandyBot™ power and disconnecting the RJ11 cable between HandyBot™ and Serial interface board.** See below:



HandyBot™ is ready to dance and sing. Place the HandyBot™ in a safe place with 1m-1.5m space diameter. Turn on the power, you will see **“Robot, Dance Press START ...”** on HandyBot™ LCD screen . Press START button on Handyboard, and wait a few seconds.

What do you see...? dancing?! Singing?!

The whole performance will take place for a minute.

#### **Example code #1: dance.c**

```
char pp_song[] = "1#d 4e3r1#f4g3r1#d 3e1#f3g1c3bD1e3g1b 8&b2b2a2g2e2d10e 7r1#d  
4e3r1#f4g3r1#d 3e1#f3g1c3b1g3b1e 28&e D3r1#d 4e3r1#f4g3r1#d  
3e1#f3g1c3bD1e3g1b 8&b2b2a2g2e2d10e 12r U3e1d3b1a3g1#f 1&b3a1&b3a1&b3a1&b3a  
2g2e2d20e" ;  
  
char w_song[] = "1c2f1f2f1g2a1a2a1f2g1g2g1e4f1r  
1f2a1a2a1&b2c1c2c1c2d1c2&b1a4g1r 1gU2d1d2d1c2d1e2f1d2c1d2c1&b4a1r  
D1c2f1f2f1g2a1a2a1f2g1g2f1e5f" ;
```

```

char classics_song[]="4g2c2d4c4g4a2g2f2e2c2d2e2f2g2a2b10c2r
4c4b4c2d2b2c2d2e2#f4g2#f2e4d2g2#f4g4d4e2d2c2b2g2a2b2c2b2c2d2e2d2e2#f12g";

char charge_song[]="1c1e1g2c1r1g4c";

charlooney_tune_song[]="U3e1d2c2d2e2d2e2c2d2d2d6d2r3d1c2b2
c2d2#c2d2b2c2c2c6c";

int tempo= 12;

long time, newtime;

int music_current_command= 'o';

int music_next_command= 0;

void main(void)
{
    int i;
    printf("Robot, dance    Press START ...\n");
    start_press();          /* waiting user to press start button */
    sleep(2.0);             /* wait for 2 seconds to start */
    charge();

    for (i=0; i<5; i++)    /* five rounds */
    {
        moveClockwise(2.0);    /* dance patterns */
        moveAntiClockwise(2.0); /* you are free to change it !!!!! */
        moveForward(1.0);
        moveClockwise(2.0);
        moveAntiClockwise(2.0);
        moveBackward(1.0);
    }
    alloff();
    printf("Bye..Bye!!      \n");
    pp();
}

void moveForward(float timeSec)
{
    beep();
    motor(0,100);            /* turn on port 0 motor with full speed */
    motor(1,100);            /* turn on port 1 motor with full speed */
    sleep(timeSec);          /* move forward for ? seconds */
}

void moveBackward(float timeSec)
{
    beep();
    motor(0,-100);           /* reverse port 0 motor with full speed */
    motor(1,-100);           /* reverse port 1 motor with full speed */
    sleep(timeSec);          /* move backward for ? seconds */
}

void moveClockwise(float timeSec)
{
    beep();
    motor(0,-100);           /* turn on port 0 motor with full speed */
    motor(1,100);            /* turn on port 1 motor with full speed */
    sleep(timeSec);          /* rotate one direction for ? seconds */
}

```

```

}

void moveAntiClockwise(float timeSec)
{
    beep();
    motor(0,100);          /* turn on port 0 motor with full speed */
    motor(1,-100);         /* turn on port 1 motor with full speed */
    sleep(timeSec);        /* rotate another direction for ? seconds */
}

void pp()
{
    play(pp_song);
}

void play(char song[])
{
    int i, duration, accidental, delta, note, rest;
    int notes[] = {0,2,3,5,7,8,10};
    int old_note = 30;
    play_reset();

    i = 0;
    while (song[i]) {
        while (1) {
            while (song[i] == ' ') ++i;
            if (song[i] == 'X') {++i; music_next_command = song[i]; ++i;}
            else break;
        }

        if (!song[i]) break;

        while (song[i] == 'D') {
            old_note -= 12;
            ++i;
        }
        while (song[i] == 'U') {
            old_note += 12;
            ++i;
        }
        if ('0' <= song[i] && song[i] <= '9') {
            duration = 0;
            while ('0' <= song[i] && song[i] <= '9') {
                duration = duration * 10 + song[i] - '0';
                ++i;
            }
        }
        if (song[i] == '#') {
            accidental = 1;
            ++i;
        } else if (song[i] == '&') {
            accidental = -1;
            ++i;
        } else {
            accidental = 0;
        }

        if (song[i] == 'r') rest = 1;
    }
}

```

```

        else {
            if (song[i] < 'a' || song[i] > 'g') {
                printf("\nBad note:%c\n", song[i]);
                beep();
                beep();
                beep();
                sleep(5.0);
                return;
            }
            note= notes[song[i]-'a'] + accidental;
            rest= 0;
        }
        i++;

        if (rest) {
            play_note(0, duration);
        } else {
            delta= note - (old_note % 12);
            old_note += delta;
            if (delta > 5) old_note -= 12;
            if (delta < -5) old_note += 12;

            play_note(old_note, duration);
        }
    }
    play_note(0, 1);
    music_command('o');
}

void music_command(int c)
{
    music_current_command= c;
}

void charge() {
    play(charge_song);
}

void play_reset()
{
    time= mseconds();
    newtime= time+100L;
}

void play_note(int note, int duration)
{
    play_note_2(note, duration*7);
    play_note_2(0, duration);
}

void play_note_2(int note, int duration)
{
    float freq;
    int period;

    if (note) {
        freq= 55.0 * (2. ^ (((float) note) / 12.));
    }

    while (mseconds() < newtime);
}

```

```

    if (note) {
        set_beeper_pitch(freq);
        beeper_on();
    }
    else {
        beeper_off();
    }
    if (music_next_command &&
        music_current_command != music_next_command){
        music_command(music_next_command);
        music_next_command= 0;
    }

    newtime += (long)(duration * tempo);
}

```

### 3.2 What did you learn from the program “dance.c?”

The program’s function name called main(..) and this is a start-up function call when the HandyBot™ is firstly being powered up.

Now, change the main(..) function name to mainTest(..) in the “dance.c.” and save it. Turn off the HandyBot™ power and reconnect the RJ11 back to it. Turn on the power again. Unloading this program by typing-in “**unload dance.c**” on IC prompt to remove the program from the HandyBot™.

**WARNING: Always unloads unused program from the HandyBot™ to save memory space and function call confusion.**

Load the modified “dance.c” program back to HandyBot™ by typing-in “**load dance.c**” Turn off the power and disconnect the RJ11 cable from the Serial interface board. Turn on the HandyBot™ power again, and wait a few seconds. HandyBot™ is not responding! Why?

This is why we called “*Interactive*” C. Since the default start-up function call is named as main(..), the modified program “dance.c” does not have this function, and the HandyBot™ will not start during power up.

Turn off the HandyBot™ power and re-connect RJ11 cable back to HandyBot™. Turn on the HandyBot™ again. Type-in “testMain()” to invoke the modified program.

Wait a few seconds. Does the LCD display show the similar display like the original program?

This “*Interactive*” feature is very useful to test individual function before combining all functions together. By using START and STOP buttons, you can create a program not using main(..) function name call. To invoke the program, just simply press START button, and pressing STOP button to end the program.

**WARNING: User’s program is stored in Static RAM (SRAM) location. The MIT Handyboard system has a battery protected SRAM even the HandyBot™ is powered off. The program will remain in the memory unless unloading the program, running out the battery power, or removing the battery power. You have to reload the Pcode and your program again when the power is lost.**

You can now work your own way.....

Before attempting your first program, please read the “Interactive-C programming manual” and “Handyboard Technical Reference Manual” located in disk 1 and 2.

It gives you a great idea about robotic control programming technique and sensor interface.

## 4.0 How to interface the provided sensor?

Good question. It depends how your robot behaves and the course objectives. Provided with the kit, you can attach the sensor using double side foam tape, also hard wiring sensors on HandyBot™ is possible because the body of the HandyBot™ can be drilled to attach other circuit sensor into it.

Here are some references about sensor interface design:

1. Handyboard Technical Reference Manual, 2000 November 15.
2. Handyboard official book “Robotic Exploration” – by Dr. Fred Martin, ISBN 0-13-089568-7. It is available from Amazon.com or RobotStoreHK.com
3. Official Handyboard site for hardware and software resources:  
<http://www.handyboard.com>
4. Handyboard based robot design:  
<http://www.robotstorehk.com/robot.html>

## 5.0 FAQ

5.1 Can I use rechargeable battery on HandyBot™ system?

Answer: Yes. You can use alkaline or rechargeable batteries to power up HandyBot™.

5.2 Why my program disappears when I replace new batteries?

Answer: The Pcode and user program resides in SRAM You need to reload the Pcode into HandyBot™ again.

5.3 Will my program lost if I turn off the power of HandyBot™?

Answer: No. The SRAM on HandyBot™ is battery packed. The Pcode and user program will not be erased unless unloading the program, running out the battery power, or removing the battery power.

5.4 Everything I issue an IC motor function call motor(0,100), the motor supposed to move forward. But now, it rotates in reverse direction. Why?

Answer: You must connect the motor in reverse direction. DC motor does not have polarity. You just need to reverse the motor connection connected to HandyBot™.

5.5 After carrying lots of robotic experiments, I am running out the programming space. How can I reduce the memory space?

Answer: Good programming technique may solve the problem. Also, unloading unused program from the HandyBot™ will be helpful. Just simply type-in “unload filename.c” to erase the file from the HandyBot™ memory.

## 6.0 Support and Contact Information

Softcopy of this document is available in:

<http://www.robotstorehk.com/handybotV20.pdf>

If you have technical difficulty or questions or comments or unclear regarding HandyBot™ or documents, please send us an email. We want to hear your comments and feedback.

Technical Support: [support@robotstorehk.com](mailto:support@robotstorehk.com)